Coq-Metatheory for Smol-Zooid

A Gentle Adventure Mechanising Message Passing Concurrency Systems, Act 3

David Castro-Perez, Francisco Ferreira, **Lorenzo Gheri**, Martin Vassor, and Nobuko Yoshida

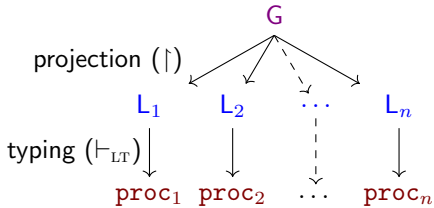
DisCoTec 2022
Lucca, 13-06-2022

Imperial College London

University of Kent

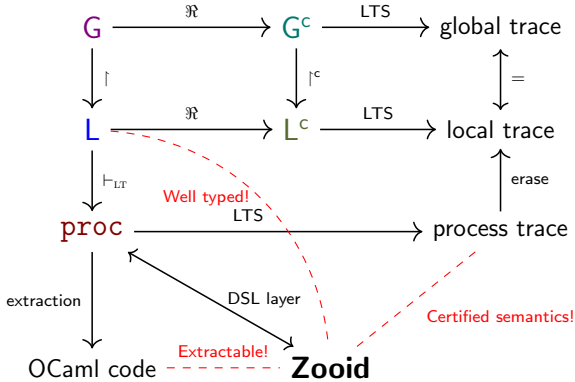
ROYAL HOLLOWAY UNIVERSITY OF LONDON

The MPST World, as We Know It



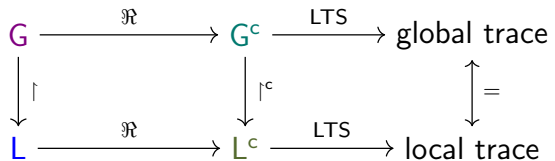
K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. POPL '08

Zooid



D. Castro-Perez, F. Ferreira, L. Gheri, and N. Yoshida. Zooid: a DSL for certified multiparty computation: from mechanised metatheory to certified multiparty processes. PLDI 2021

Introducing the Metatheory of Smol-Zooid Types



- unravelling preserves projection; focus on coinduction (1^{st} square)
- trace equivalence; focus on soundness (2^{nd} square)

`https://github.com/emtst/GentleAdventure/act3`

Formalisation of Global and Local Types

Inductively Defined Datatypes

$$
\begin{aligned}
\mathsf{G} \;::=\; & \mathtt{end} \\
& \mid X \\
& \mid \mu X.\mathsf{G} \\
& \mid \mathsf{p} \to \mathsf{q} :(\mathbb{S}).\mathsf{G}
\end{aligned}
$$

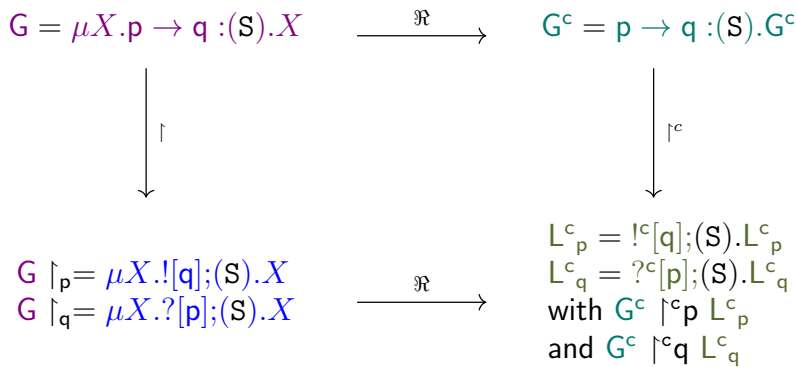
$$
\begin{aligned}
\mathsf{L} \;:=\; & \mathtt{end} \\
& \mid X \\
& \mid \mu X.\mathsf{L} \\
& \mid \,!\,[\mathsf{q}];(\mathbb{S}).\mathsf{L} \\
& \mid \,?\,[\mathsf{p}];(\mathbb{S}).\mathsf{L}
\end{aligned}
$$

Coinductively Defined Datatypes

$$
\begin{aligned}
\mathsf{G}^{\mathsf{c}} ::= \; & \mathtt{end}^{\mathsf{c}} \\
& \mid \mathsf{p} \to \mathsf{q} :(\mathbb{S}).\mathsf{G}^{\mathsf{c}} \\
& \mid \mathsf{p} \rightsquigarrow \mathsf{q} :(\mathbb{S}).\mathsf{G}^{\mathsf{c}}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{L}^{\mathsf{c}} ::= \; & \mathtt{end}^{\mathsf{c}} \\
& \mid \,!^{\mathsf{c}}[\mathsf{p}];(\mathbb{S}).\mathsf{L}^{\mathsf{c}} \\
& \mid \,?^{\mathsf{c}}[\mathsf{q}];(\mathbb{S}).\mathsf{L}^{\mathsf{c}}
\end{aligned}
$$

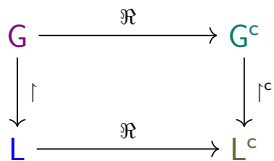
Formalisation of Global and Local Types



Abandoning Inductive Datatypes

Theorem (Unravelling preserves projections)

Given G, L, G^c *and* L^c, *such that* (a) $\mathsf{G}\restriction_{\mathsf{r}} = \mathsf{L}$, (b) $\mathsf{G}\mathrel{\Re}\mathsf{G}^c$, *and* (c) $\mathsf{L}\mathrel{\Re}\mathsf{L}^c$, *then* $\mathsf{G}^c \restriction^c_{\mathsf{r}} \mathsf{L}^c$.



Proof.

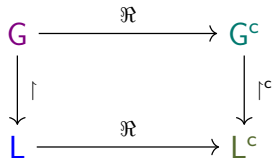
By coinduction. :) □

The Paco Library for Coq: https://plv.mpi-sws.org/paco/

Abandoning Inductive Datatypes

Theorem (Unravelling preserves projections)

Given G, L, G^c *and* L^c, *such that* (a) $\mathsf{G}\restriction\mathsf{r} = \mathsf{L}$, (b) $\mathsf{G}\Re\mathsf{G}^c$, *and* (c) $\mathsf{L}\Re\mathsf{L}^c$, *then* $\mathsf{G}^c \restriction^c \mathsf{r}\ \mathsf{L}^c$.



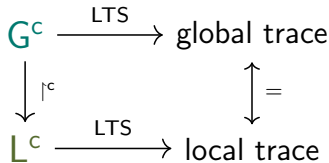
⟶ Coq!

Proof.

By coinduction. :) □

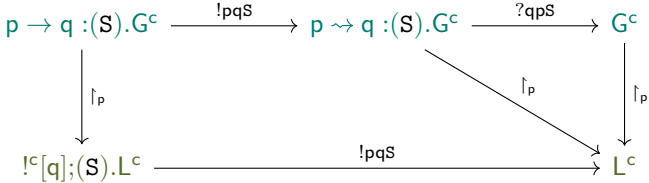
The Paco Library for Coq: https://plv.mpi-sws.org/paco/

Type Semantics for Zooid

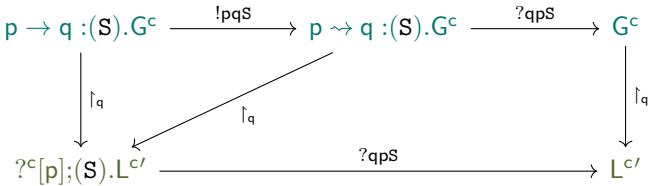


With Love, from p to q

p sends:



q receives:



Tools for our LTS

Actions. $!\mathsf{pqS}$ and $?\mathsf{qpS}$

(Local) Environments. E such that, $E(\mathsf{p}) = \mathsf{L}^{\mathsf{c}}{}_{\mathsf{p}}$ where $\mathsf{G}^{\mathsf{c}} \upharpoonright^{\mathsf{c}} \mathsf{p}\ \mathsf{L}^{\mathsf{c}}{}_{\mathsf{p}}$

Queues and Queue Environments. Q, buffers for asynchronous communication.

$$!^{\mathsf{c}}[\mathsf{q}];(\mathsf{S}).\mathsf{L}^{\mathsf{c}} \xrightarrow{\text{step}} \mathsf{L}^{\mathsf{c}}$$

$$Q(\mathsf{p},\mathsf{q}) = [] \xrightarrow{\text{enqueue}} Q(\mathsf{p},\mathsf{q}) = [\mathsf{S}] \xrightarrow{\text{dequeue}} Q(\mathsf{p},\mathsf{q}) = []$$

$$?^{\mathsf{c}}[\mathsf{p}];(S).\mathsf{L}^{\mathsf{c}\prime} \xrightarrow{\text{step}} \mathsf{L}^{\mathsf{c}\prime}$$

Tools for our LTS

Actions. $!\mathsf{pqS}$ and $?\mathsf{qpS}$

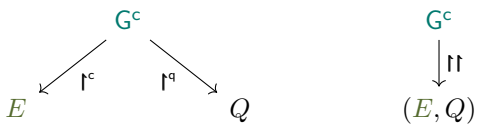
(Local) Environments. E such that, $E(\mathsf{p}) = \mathsf{L}^{\mathsf{c}}{}_{\mathsf{p}}$ where $\mathsf{G}^{\mathsf{c}} \upharpoonright^{\mathsf{c}} \mathsf{p}\ \mathsf{L}^{\mathsf{c}}{}_{\mathsf{p}}$

Queues and Queue Environments. Q, buffers for asynchronous communication.

$$!^{\mathsf{c}}[\mathsf{q}];(\mathsf{S}).\mathsf{L}^{\mathsf{c}} \xrightarrow{\text{step}} \mathsf{L}^{\mathsf{c}}$$

$$Q(\mathsf{p},\mathsf{q}) = [] \xrightarrow{\text{enqueue}} Q(\mathsf{p},\mathsf{q}) = [\mathsf{S}] \xrightarrow{\text{dequeue}} Q(\mathsf{p},\mathsf{q}) = []$$

$$?^{\mathsf{c}}[\mathsf{p}];(S).\mathsf{L}^{\mathsf{c}\prime} \xrightarrow{\text{step}} \mathsf{L}^{\mathsf{c}\prime}$$



Theorems

Theorem (Step Soundness)

If $\mathsf{G}^c \xrightarrow{a} \mathsf{G}^{c\prime}$ *and* $\mathsf{G}^c \upharpoonright \lceil(E, Q)$*, there exist* E' *and* Q' *such that* $\mathsf{G}^{c\prime} \upharpoonright \lceil(E', Q')$ *and* $(E, Q) \xrightarrow{a} (E', Q')$*.*

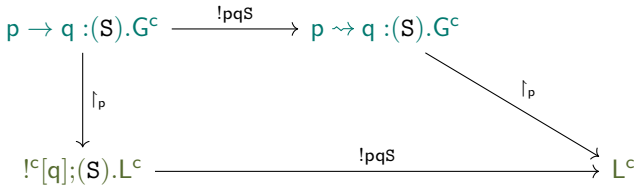
Theorem (Step Completeness)

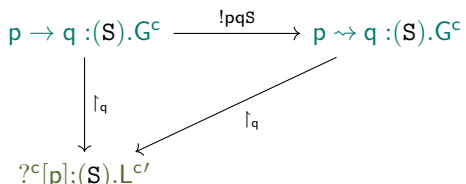
If $(E, Q) \xrightarrow{a} (E', Q')$ *and* $\mathsf{G}^c \upharpoonright \lceil(E, Q)$*, there exist* $\mathsf{G}^{c\prime}$ *such that* $\mathsf{G}^{c\prime} \upharpoonright \lceil(E', Q')$ *and* $\mathsf{G}^c \xrightarrow{a} \mathsf{G}^{c\prime}$*.*

Theorem (Trace equivalence)

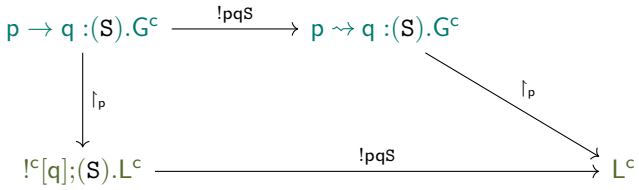
If $\mathsf{G}^c \upharpoonright \lceil(E, Q)$*, then* $\mathtt{tr}^g t \mathsf{G}^c$ *if and only if* $\mathtt{tr}^l t(E, Q)$ *.*

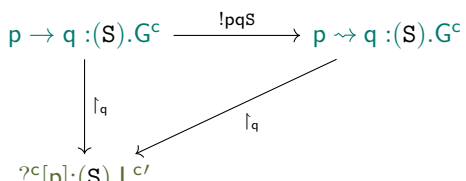
Lemma, to give the flavour





Lemma, to give the flavour





$\longrightarrow$ Coq!

You Suffer...

- Formal proofs are not easy.
- Proof design is the key.
- Proof techniques are to be taken seriously: (co)induction, functions VS relations, treatment of bindings...

→ D. Castro-Perez, F. Ferreira, L. Gheri, and N. Yoshida. "Zooid: a DSL for certified multiparty computation: from mechanised metatheory to certified multiparty processes". PLDI 2021.
DOI: `https://doi.org/10.1145/3453483.3454041`
website: `http://mrg.doc.ic.ac.uk/publications/zooid-paper/`

→ This tutorial is available at `https://github.com/emtst/GentleAdventure`

... but Why?

Formal proofs are not easy

[1] Aydemir et al. "Mechanized Metatheory for the Masses: The POPLmark challenge." 2005

... but Why?

Formal proofs are not easy, but useful and fun!

As witnessed, e.g., by the influential POPLmark Challenge[1]...

[1] Aydemir et al. "Mechanized Metatheory for the Masses: The POPLmark challenge." 2005

... but Why?

Formal proofs are not easy, but useful and fun!

As witnessed, e.g., by the influential POPLmark Challenge[1]...

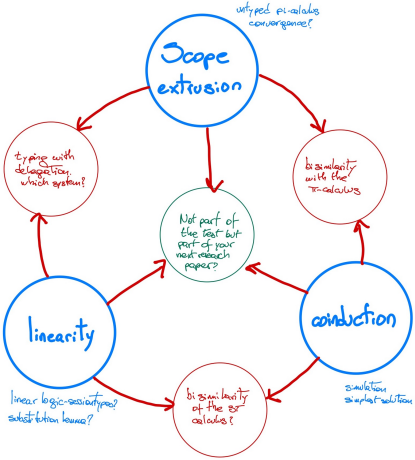
Towards a **Concurrent Calculi Formalisation Benchmark**

Challenge problems:

- name passing and scope extrusion
- linearity and behavioural type systems
- coinduction and reasoning about process algebras

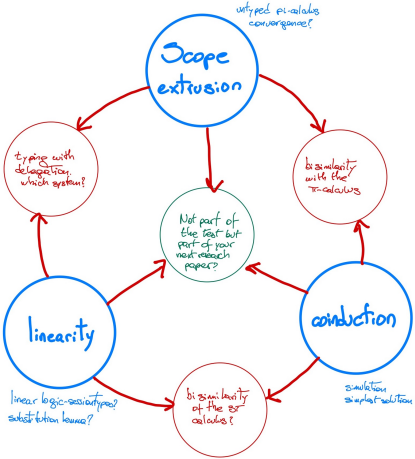
[1] Aydemir et al. "Mechanized Metatheory for the Masses: The POPLmark challenge." 2005

The Future is Unwritten... But Sketched!



- Concurrent Benchmark website: `https://concurrentbenchmark.github.io/`
- This tutorial: `https://github.com/emtst/GentleAdventure`

The Future is Unwritten... But Sketched!



- Concurrent Benchmark website: `https://concurrentbenchmark.github.io/`
- This tutorial: `https://github.com/emtst/GentleAdventure`

THANK YOU!